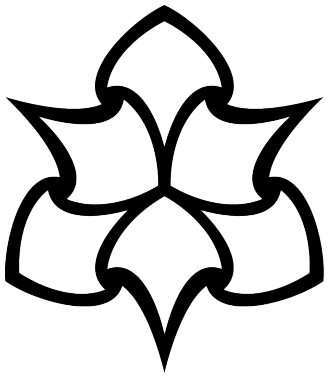


A Flexible Encryption Technique for the Internet of Things Environment

S. Medileh[a], A. Laouid[a,b,*], E-B. Nagoudi[a], R. Euler[e], A. Bounceur[e], M. Hammoudeh[d], M. AlShaikh[c], A. Eleyan[d], Osama A. Khashan[c]

[a] *University Echahid Hamma Lakhdar, PO Box 789, 39000 El Oued, Algeria*
[b] *LIMED Laboratory, University of Bejaia, 06000, Bejaia, Algeria*
[c] *Saudi Electronic University, KSA.*
[d] *School of Computing, Mathematics and Digital Technology Manchester Metropolitan University, United Kingdom*
[e] *Lab-STICC UMR CNRS 6285, Université de Bretagne Occidentale, 20 Avenue Victor Le Gorgeu, 29238 Brest, France*

Abstract

IoT promises a new era of connectivity that goes beyond laptops and smart connected devices to connected vehicles, smart homes, smart cities and connected healthcare. The huge volume of data that is collected from millions of IoT devices raises information security and privacy concerns for users. This paper presents a new scalable encryption technique, called Flexible encryption Technique (FlexenTech), to protect IoT data during storage and in transit. FlexenTech is suitable for resource constrained devices and networks. It offers a low encryption time, defends against common attacks such as replay attacks and defines a configurable mode, where any number of rounds or key sizes may be used. Experimental analysis of FlexenTech shows its robustness in terms of its multiple configurable confidentiality levels by allowing various configurations. This configurability provides several advantages for resource constrained devices, including reducing the encryption computation time by up to 9.7% when compared to its best rivals in the literature.

Keywords: Internet of Things, Encryption, Security.

1. Introduction

The Internet of Things (IoT) is a set of perceptible connected devices capable of interacting with each other. A *thing* refers to any device connected to the physical and digital world over the Internet [1]. These things progressively compose an embedded system that serves several real world environments like healthcare and intelligent transport. IoT devices embrace physical objects via digital actuators, RFID tags, sensors and communication units. IoT devices are often characterized by ultra-low bandwidth and limited computation and communication capabilities. As shown in Figure 1, most of the IoT scenarios are looking to connect with everything and everyone in order to share information [2]. With the huge number

*Corresponding author
Email address: abdelkader-laouid@univ-eloued.dz (A. Laouid)

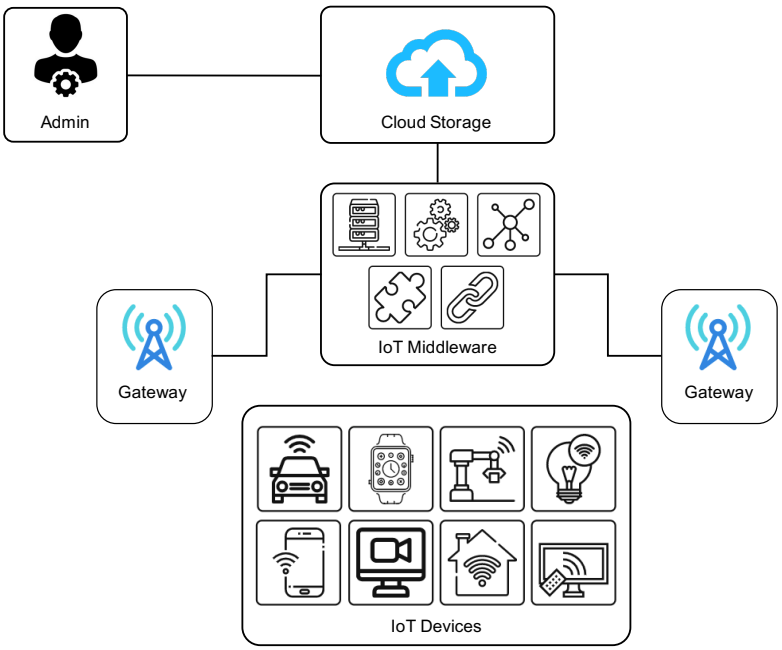


Figure 1: A comprehensive vision of the IoT architecture.

of connected entities the network traffic as well as the storage capacity will increase in an exponential way. This introduces critical security challenges since efficient encryption, e.g., public key infrastructure, will not operate reliably. These limitations render IoT devices prone to a wide range of attacks. Cloud computing offers a solution to overcome the resource constraints of IoT devices. Furthermore, offloading data and outsourcing computational tasks to the cloud presents new security and privacy concerns. The amount of private information generated by IoT devices intensifies the security and privacy challenges in cloud-enabled IoT networks. As the adoption of IoT grows, the cost-benefit of designing rigorous protocols will become a major research subject with great impact. Hence, IoT applications cope with many challenges, particularly the difficulties related to privacy and security issues [3]. These challenges include low resource availability issues, vulnerability problems due to the nature of wireless communication, and low-power computation resources. These constraints explain the difficulty of developing the necessary protocols for the management and protection of these networks, such as routing and security protocols[4]. Like the other types of network architectures, security is one of the most challenging requirements in IoT networks [5] which includes mainly data confidentiality, data integrity, authentication, availability, and data freshness. Any adapted IoT architecture and its standard protocols must consider essential restrictions such as dynamicity, reliability, scalability, etc., and the information exchanged between IoT devices and the cloud must be encrypted with adequate security prior to transmission.

Most of the classical cryptographic systems and algorithms are not suitable for IoT networks as they

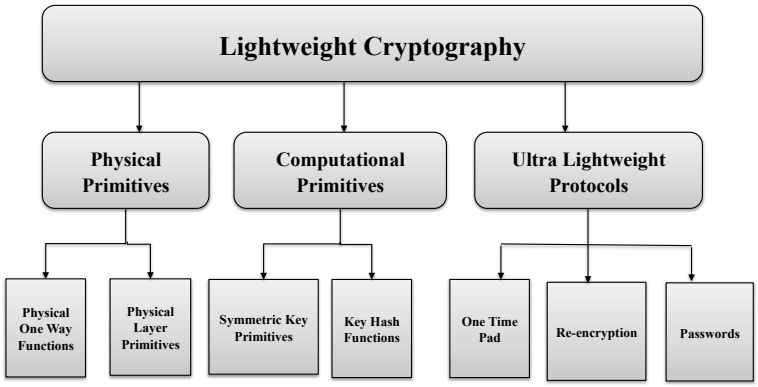


Figure 2: A Lightweight Cryptography Taxonomy.

are designed for resource-rich devices on high bandwidth networks. To address this issue, we propose a lightweight block cipher that consumes fewer resources and reduces encryption overhead by realising: smaller block size, smaller key length, lower computational complexity and simple key generation. Such a cipher depends on the specifically used hardware technology and coding style which are useful to provide an accurate IoT application. The critical issues that must be addressed in this cipher are: how to secure inter-device and device-to-cloud communications, how to ensure the security and confidentiality of various network entities, and what are the techniques adopted for authentication, access control and other security features?

2. Background and related work

This section aims to present an overview of lightweight cryptography techniques and to discuss some existing work. The proposed lightweight cryptography techniques will be explained depending on their characteristics.

2.1. A taxonomy of lightweight cryptography techniques

The main characteristics of lightweight cryptography are low-cost, performance and applicable security level. An optimal lightweight cryptography has to achieve a good balance between security level, computation time and power consumption.

Regarding these characteristics, we can classify the previous work in lightweight cryptography into three main classes: physical primitives, computational primitives and ultra lightweight protocols as shown in Figure 2.

2.1.1. Physical primitives

The main principle of physical primitives is the use of bio-metrics for authentication. A physical primitive can extend an analogue singularity or variation in physical features, which is essential to physical structures

but excessively hard to duplicate, and translate it into a digital value for the determination of a specific quantification. Physical layer protocols are designed with the aim to minimise power consumption without compromising the fidelity. Power consumption should be linearly scaled as a function of data rate [6]. In [7], the authors describe Physical One-Way Functions (POWF) which focus on measuring the accuracy of sprinkling outlines of noticeable laser energy. This approach provides a low implementation cost, but it is not an appropriate low-cost explanation for radio-frequency identification (RFID) because if one delay is consistently greater than another across chips, then the response will always be the same.

Other authors, e.g., [8], used POWF in their work to decrease the cost of implementation but it reduces the performance.

2.1.2. Computational primitives

In computational primitives, the system security relies on the complexity in term of computation. These approaches provide complicated mathematical solutions with high security level. In particular, asymmetric cryptographic primitives have been approved as a strong solution to protect the data or secure the communication since every primitive has been considered a secure entity [9]. In asymmetric cryptography, computational primitives exploit the basics of modular arithmetic to attain Shannon's philosophies of confusion and diffusion [10] with expending basic mathematical and logical processes. In [11, 12, 13, 14], hash functions and symmetric cryptography schemes are presented. The main shortfalls of these approaches are the use of large keys, high complexity and energy consumption assigned to the computation process. In addition, these approaches have not been investigated by public examination to experiment the security aspect.

2.1.3. Ultra Lightweight Protocols

Ultra lightweight refers to those security approaches that use simple logic operations such as exclusive-or (XOR) for their application [15, 16]. These kinds of techniques become popular and more adaptive through RFID technologies. Indeed, their security aspect can provide the main services of the security resistant. These schemes focus on a set of applied assumptions, where the attacker will not have the facility to compute and generate the key. However, some of them consider that the attacker can read the tag for a too limited time slot. Ultra lightweight primitives can be classified into one-time pads, as in [17, 18], which are based on pseudonyms, re-encryption [14] and configured passwords [19].

As a result, secret keys are no longer essential to be securely stored in the memory of physical primitives. These techniques usually reduce the cost of implementation, as well as the security performance. The focus of computational primitives on complicated mathematical solutions makes them suffer from a security problem, in particular from key moving between the sender and receiver. A well designed technique may render them more popular and adaptive for low-end connected devices.

2.2. Related work

Numerous research works have been recently carried out to provide encryption techniques pertinent to the IoT context. They address the inherent resource limitations of IoT devices that make it difficult to apply conventional encryption algorithms that require a significant amount of resources for their operation [20]. Several researchers cope with the resource limitation challenge by proposing lightweight encryption techniques to provide efficient and secure communication with reasonable resource utilisation. A number of researchers devised techniques to optimise existing conventional block ciphers for IoT devices. Some of them have improved the performance of block ciphers by utilising smaller key sizes [21], reducing the number of encryption rounds [22] and using smaller block sizes [23].

Other researchers focused on key management and authentication challenges in IoT environments [24]. The vast majority of this research focuses on symmetric-based algorithms. For example, Beaulieu et al. [25] proposed Simon and Speck which are lightweight block ciphers with a variety of block and key sizes. The work in [26] proposed a model which uses parallel computation to enhance the performance of the block cipher algorithm. This model has been designed to consume lower energy and reduce hardware complexity. The SIT algorithm proposed in [27] focuses on a hybrid Feistel and substitution-permutation network (SPN) architecture. It is a mixture that uses the advantages of both approaches to develop a lightweight algorithm that provides substantial security while maintaining the computation complexity at a reasonable level in an IoT environment. The symmetric encryption scheme proposed by [28] uses multiple chaotic dynamical systems for the IoT domain. This technique maintains the dynamic key update with each input data block to provide higher levels of randomness. The authors in [29] present an experimental configuration of a fractional-chaos based-cryptosystem for an IoT-based architecture in ad hoc networks under the IEEE 802.15.4 standard. The solution benefits from the characteristics of the fractional-order derivative operator for encryption schemes to provide secure communications.

Lightweight asymmetric based solutions may provide stronger security than symmetric ciphers. However, asymmetric solutions are not highly scalable and often have higher computation complexity which renders them unsuitable for IoT environments [28]. Recently, several studies such as Identity-Based Encryption (IBE) [30, 31] have been published which combine user's identity with series of attributes to encrypt data while enforcing a secure access policy. Furthermore, the decryption is carried out when authorized users with the desired attributes satisfy a threshold access control policy. In [15], the authors proposed a lightweight IBE scheme by using pre-computation techniques to reduce the computation cost of constrained devices. They designed a lookup table, which obtains a pre-computed set of pairs generated using elliptic curve cryptography in order to be used later to carry out cryptographic operations at low computational cost. To overcome the expensive bilinear pairing computations, the authors in [16] proposed a lightweight no-pairing IBE scheme based on elliptic curve cryptography by replacing pairing operations with point scalar multiplication.

In [32], the authors proposed a lightweight encryption scheme for smart homes. This scheme provides users and smart objects a secure service at the lowest computation and communication costs. The proposed solution relies on identity-based encryption to support flexible public key management, which does not require complex certificate handling.

Attribute-Based Encryption (ABE) provides the necessary flexibility required to manage a large number of IoT devices and fine-grained access control, which is very much suited to secure data transmission, storage and sharing in cloud-based IoT systems. The authors in [15, 33] give background information on the Ciphertext-Policy Attribute-Based Encryption (CP-ABE) and pre-computation techniques in an efficient and secure way, and then describe the extended ABE scheme proposed for the CloudIoT platform [34, 33].

In [35], Alphand et al. propose a blockchain security architecture for IoT named *IoTChain*. In this work, the secure authorized access to IoT resources is ensured by a combination of the ACE authorization framework [36] and the OSCAR architecture [37]. IoTChain replaces the single ACE authorization server by the blockchain to make the ACE authorization phase flexible and trustless. The feasibility of IoTChain is evaluated through an implemented authorization blockchain on top of a private Ethereum network. Dinu et al. [38] introduced a software benchmarking framework for the consistent evaluation of lightweight block ciphers on three different platforms, namely 32-bit ARM, 16-bit MSP430 and 8-bit AVR. Recently, a Novel Tiny Symmetric encryption Algorithm (NTSA) has been presented in [39]. NTSA proposes to enhance the text file transfer security through the IoT network by introducing additional key confusions dynamically for each round of encryption.

2.3. Summary and contribution

As a conclusion, all these works might provide a satisfying solution to secure the communication. However, there are still some inherent limitations in terms of efficiency, scalability and access control, which make them not suitable for constrained IoT devices. Therefore, there has been a growing demand for efficient lightweight encryption mechanisms that combine all the features of lightweight symmetric and asymmetric algorithms [28]. Furthermore, these mechanisms should be able to provide mutual lightweight authentication for secure access control and authentication between users and devices in an IoT environment. In this paper, we will design and implement a lightweight encryption technique, which is targeted to define a secure information exchange by considering the encountered limitations. To this end, an autonomous and secure exchange of confidential information in the IoT context is required in regard to the nature of the connected entities and their real environment. The proposed technique focuses on distributed architecture networks with the aim to ensure a high security on the basis of a dynamic topology.

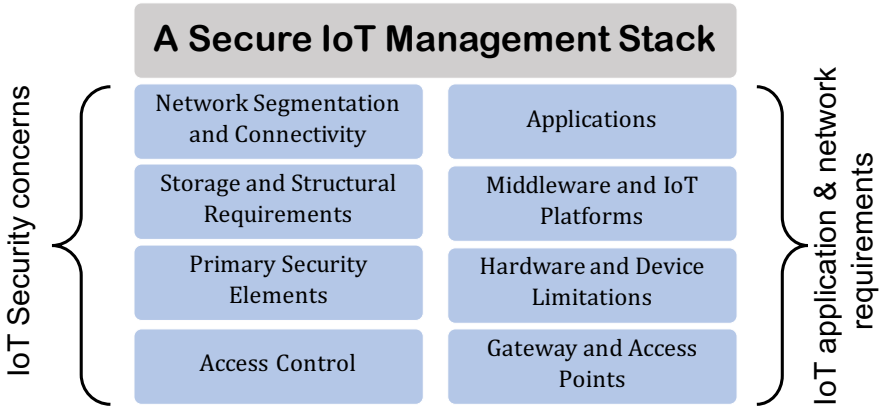


Figure 3: A Secure protocol stack for IoT.

3. Architectural concept

In this section, we present the main requirements for a secure information exchange that considers the limitations of IoT devices. Then we describe the proposed architecture that addresses these requirements. Our work focuses on distributed architecture networks with the aim to ensure an adapted security level with dynamic architecture topology. The reason behind flat and distributed network architectures is to adapt to the nature of IoT networks, where several entities of connected devices may autonomously and securely exchange confidential information.

A sustainable IoT architecture cannot be achievable without protecting the information, either during the transmission or the storage. The main tasks here are protection and management. Figure 3 shows a secure IoT management stack that contains two sub-stacks, IoT having a layered communication protocol as shown in the right sub-stack, that allows an enormous number of objects to get connected through the internet. These layers are to fulfill the IoT network limitations and needs. Suitable activity scenarios could be reached when the second sub-stack is accompanied by the first. That explains why the security paradigm of IoT network architectures becomes one of the most indispensable elements. Therefore, lightweight computation modules, such as simple permutation functions and bit-wise exclusive-or operation, are required in the design of secure transmission for each proposed protocol. To this end, we propose in this work a flexible encryption technique, called FlexenTech, that provides an acceptable security level with respect to IoT limitations and requirements.

Table 1 contains the abbreviations used in this paper together with a brief explanation to speed up the reading and ease the understanding of its content.

The proposed technique was designed with the following goals in mind:

- the technique should be a symmetric block cipher. The same secret cryptographic key (key $= K$) is used for encryption and for decryption. The plain data and cipher data are fixed-length bit sequences

Acronym	Definition
K	is the secret **K**ey, each pair of entities asymmetrically shares the same secret number called K.
B	plain or ciphered data **B**lock size in term of bits.
V_i	represents the calculated **V**alue for the $\mathbf{i}^{th}$ bit position.
rnd	is the number of rounds that will be used in a given configuration.
r	the current round.
st_r	means the reversing start bit position from which the next bits will automatically be reversed.

Table 1: List of used acronyms

(block size = B).

- it should be suitable for low resource hardware. This means that the proposed technique should use only primitive computational operations.
- it should be adaptable to devices of different data-lengths. Accordingly, the number B of bits in a data is a parameter of our proposed technique; different choices of this parameter can be used.
- it should be iterative in structure, with a variable number of rounds. The number of rounds is a second parameter as shown in our architecture (Figure 4). The user can explicitly maintain the trade-off between high speed and high security.
- it should be simple and easy to implement. It is more important to use a simple structure as illustrated in Algorithm 1, with the aim to easily implement it. Furthermore, a simple scheme is perhaps more interesting to analyze and evaluate, so that the cryptographic strength can be more rapidly determined.

3.1. The proposed IoT encryption architecture

There are several emerging areas where highly constrained devices are interconnected to accomplish some tasks. These constrained devices (IoT) aim to communicate and make decisions over many low resources and capacities. To an efficient use goal, the designed conceptual model illustrated in Figure 4 shows the complexity space of the proposed technique in view of the security requirements such as confidentiality, integrity and signature. The efficiency of FlexenTech resides on the choice of B and rnd, where its flexibility offers a complete freedom to the user for weighing between the complexity and the security level. In heterogeneous environments, the FlexenTech encryption mode can respond to many challenges and issues like power consumption of devices, limited battery, memory space, performance cost, and security.

The proposed technique focuses on three parameters to encrypt/decrypt the information:

- Each pair of nodes shares a secret number which will be used as a key.

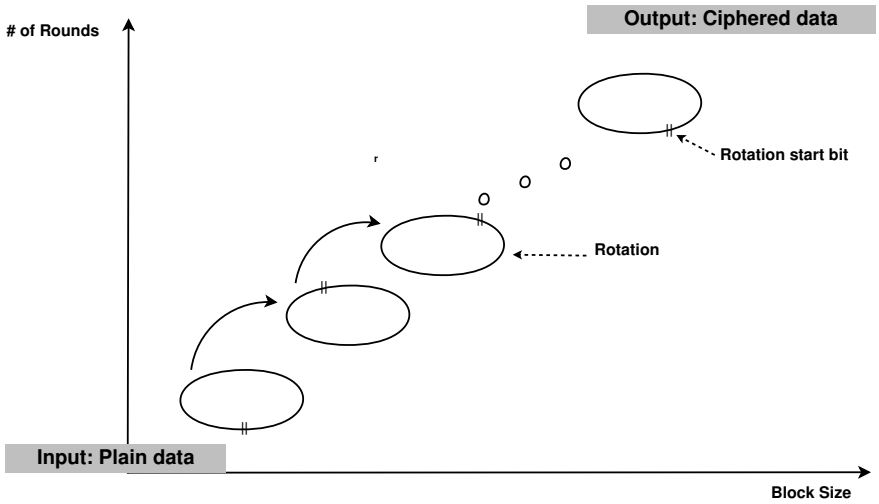


Figure 4: A conceptual representation of FlexenTech in view of the complexity space (*Block Size*, *Round*).

- A number of rounds should be carried out before achieving the final encrypted information.
- In each given round i, a random rotation will be applied to the information obtained from the previous round $i-1$.

The proposed architecture performs a set of random permutations, substitutions and rotations at the bit level to encrypt the plaintext. The permutations and substitutions are done using the modular function with the following steps:

- The size of the information to be encrypted is pre-configured with a given block size called B (in terms of number of bits).
- Using one of the proposed key management schemes, such as *Self-VKS* [40], the entities may securely share a given large number K. We note that B and K should verify the conditions $\gcd{}^{1}(K, B) = 1$ and $K > B$.
- The node uses Equation (1) to make a random permutation at the bit level of the plain information that will be sent. Figure 5 shows an example that explains how this equation produces random bit permutations.
- It is possible to construct lightweight hash functions with the aim to ensure a substitution based on local variable values. Since the modulus operation provides a new random i' for each given i, all bits between the st_r bit position and the B bit position will be reversed. Hence, this situation means that

[1] gcd(K,B) means the greatest common divisor of K and B

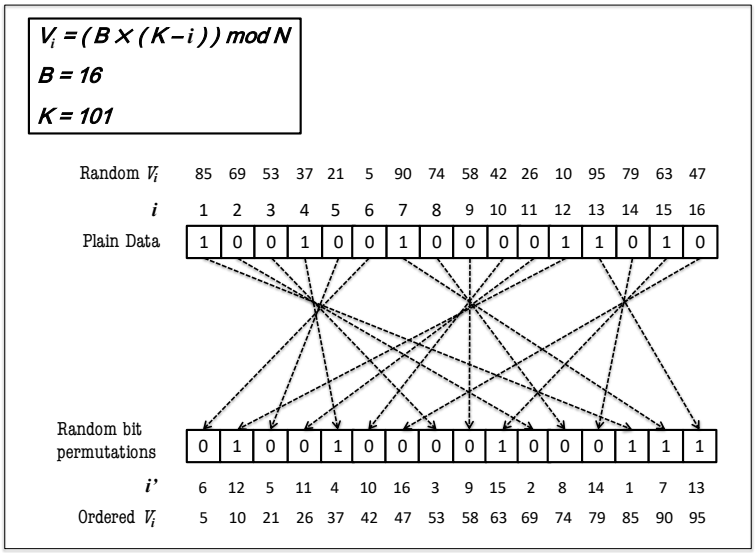


Figure 5: A demonstrative example of FlexenTech's permutation.

the content value of a given bit i will automatically be reversed if $st_r \leq i \leq B$. Otherwise, the content value will not be changed as shown in the example of Figure 7.

The key principle behind FlexenTech is that by securely sharing any given large number K, the entities can generate a set of B random values which will be used to encrypt and decrypt the exchanged information. The permutation of the bits is based on Equation (1).

$$V_i = (B \times (K - i)) \mod K \tag{1}$$

where $i = 1, 2 \ldots B$ and V_i represents the obtained value for the i^{th} bit position in the message of size B. Hereafter, the values of V_i will be ascendingly ordered with the aim to set up a random bit permutation. Every bit i will be permuted to its new position i' depending on the position of V_i in the ascendingly ordered list. To ensure that every bit will be permuted, the value of K should be selected in such a way that $\gcd(K, B) = 1$. Figure 5 shows an example of the proposed technique, where the entities securely share $K = 101$ and the size of the data is configured to $B = 16$. On the other hand, the destination node reconstructs the original information by applying Equation (1) on the received data to resubmit every bit on its right position.

3.2. The proposed FlexenTech encryption algorithm with recursive rounds

The resource consumption of the proposed security protocol must be lightweight in terms of communication, memory usage and computation. To make sure that the proposed protocol is feasible and practical, it should add a communication overhead as small as possible and have low memory consumption and low CPU workload. Therefore, an efficient encryption mechanism has been adopted. The instructions for use

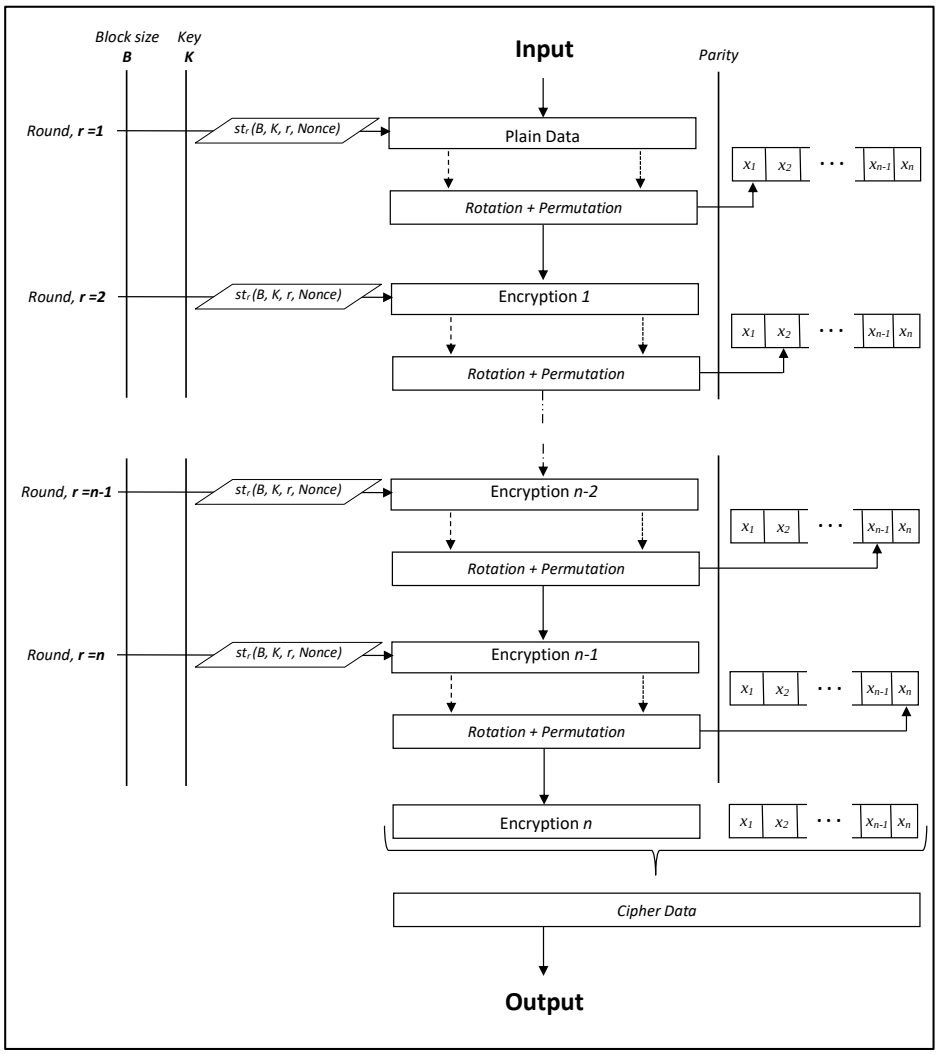


Figure 6: General structure of FlexenTech.

of this mechanism are shown in Figure 6, where a set of rounds, performing the same operations over the obtained encrypted information, should be applied before obtaining the final encrypted information. On the other hand, the proposed technique focuses on a configurable security threshold which defines the preferred security level of the communication. The security threshold is the number of required rounds to cipher a plain data. Additionally, a given threshold determines the overload size that will be added in terms of the number of additional bits to ensure data integrity and signature. Furthermore, before maintaining any round, a random rotation is performed, where the start bit of the rotation is computed by Equation (2). On the other hand, the parameter number of rounds is considered as the security level because it implies immediately the similarity rate between plain and ciphered text, the computation time, and usually an

additional overload.

$$st_r = (K \times (r + Nonce)) \mod B \tag{2}$$

where st_r is the rotation start bit of a given round r and $Nonce$ is a *nonce*, or could be a counter, which is used with the aim to reduce playback attacks. Figure 7 shows a clear example of the ciphering steps with a security level $rnd = 4$.

In the following, we explain how the proposed algorithm is applied within an IoT environment. For descriptive purposes, Algorithm 1 uses some terms defined in modular computation. An originator device prepares messages with the use of this algorithm in order to protect the data. The private data are the value of K and the secret information. The value of the data size, the used $Nonce$ and the security level, i.e., the number of rounds, can be used as public variables. The encryption table is defined to ensure a random permutation. Its size equals $3 \times B$, where the first row contains the values of V_i. The second and third rows are to save the original and the new bit position, respectively, after sorting the V_i values. For more details, *FlexenTech.c 1* shows the C code of the example illustrated in Figure 7.

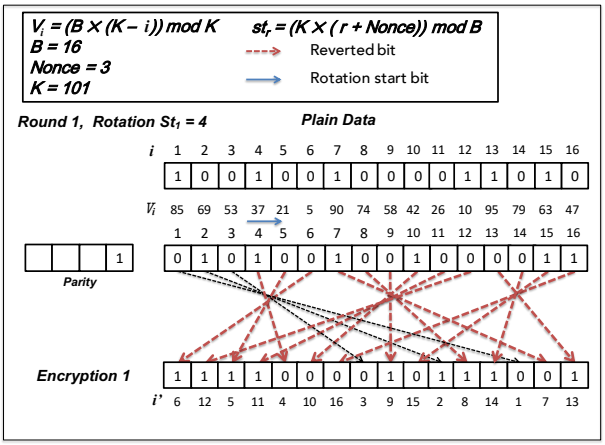
3.3. Analysis of the proposed encryption algorithm

The efficiency of any cryptographic protocol focusing on any algebraic structure depends on factors such as: parameter size, time-memory tradeoffs, available processing power, software and/or hardware optimization, and mathematical algorithms. Therefore, when designing the proposed technique we have taken into account the primary concerns to use powerful and efficient mathematical operations and algorithms which quickly carry out computations respecting at the same time the occurred overhead and the used memory storage. Several hash and encryption functions or digital signature schemes require computations in $\mathbb{Z}_m$, the integers *modulo* m (m is a large positive integer which may or may not be a prime). $\mathbb{Z}_m$ is exploited in many contexts of modern applied cryptography. In this section, we discuss the requirements and the complexity of the proposed technique. We also discuss efficient methods that can be used to perform addition and multiplication in $\mathbb{Z}_m$.

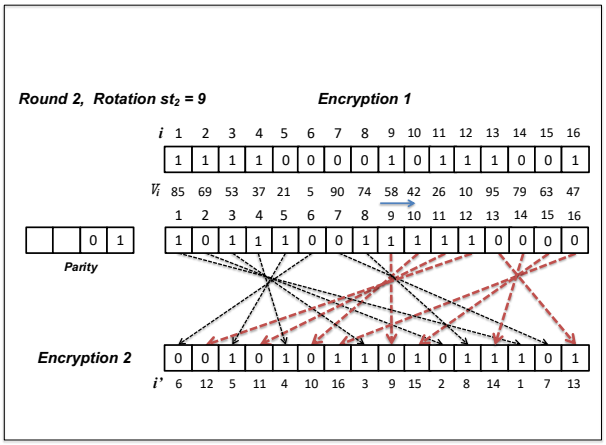
As the designed technique is bit-oriented, all of the basic computational operations have $B-bit$ as inputs and outputs. Hence, the technique is a block-cipher with a one-word of B bit input ($plainData$) block size and a one-word ($cipherData$) output block size. The choice for B is not limited to such values. Therefore, the technique is well-defined for any $B > 0$, although for strong encryption and more efficiency it is proposed to use a large value of B. We note that there is no limit for the values of B and K [41], but the relation between them should verify $\gcd(K, B) = 1$ to ensure that

$$B \times i \; mod \; K \neq 0 \; \forall \; i \in \{1, 2, \ldots B\} \tag{3}$$

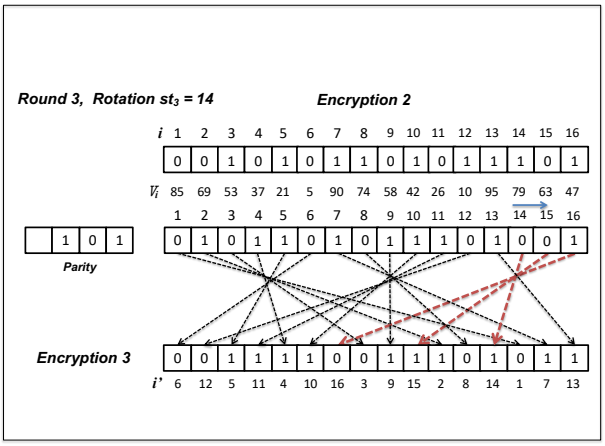
Lemma 1. *For $i \in \{1, \ldots B\}$ let $V_i \in Z_k$ with $K > 0$. If B is relatively prime to K, then the congruence $B \times i \equiv V_i \pmod{k}$ has a solution i; moreover, any integer i' is a solution if and only if $i \equiv i' \pmod{K}$.*



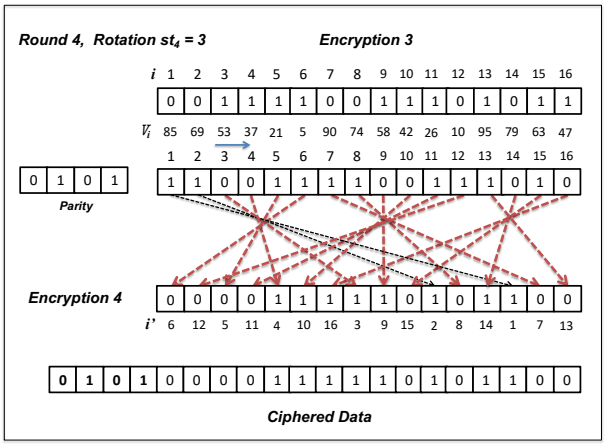
(a) Pre-configuration and steps to obtain the first round.



(b) Steps to obtain the second round.



(c) Steps to obtain the third round.



(d) Steps to obtain the fourth round and the final ciphered data.

Figure 7: A FlexenTech encryption example with four rounds.

Algorithm 1 Encryption and Decryption FlexenTech algorithm.

Require: Private: K, $plainData$;

Require: Public: B, $Nonce$, rnd, $mode$, $cipheredData$;

Ensure: $data$; ▷ $data$ here means $plainData$ or $cipheredData$ which is tied to the encryption mode.

Temporal: $table[3][B]$;
Constant: $V = 0$, $orgPos = 1$, $newPos = 2$;
for $(i \in [0, B-1])$ **do**
    $table[V][i] \leftarrow (B \times (K - i)) \mod K$;
    $table[orgPos][i] \leftarrow i$ ▷ numbering the bit positions
end for
$table \leftarrow sortingTable(table)$
if $mode = encrypt$ **then**
    $cipheredData \leftarrow plainData$
    for $r \in [1, rnd]$ **do**
        $st_r \leftarrow (K \times (r + Nonce))\ mod\ B$
        $cipheredData \leftarrow rightRotation(cipheredData,\ st_r)$;
        $cipheredData \leftarrow reverse(cipheredData,\ st_r)$;
        $parityBit(cipheredData,\ st_r)$;
        $cipheredData \leftarrow RandomPermutation(table, cipheredData)$;
    end for
    return $cipheredData$;
end if
if mode = decrypt **then**
    $plainData \leftarrow cipheredData$
    for $r \in [1, rnd]$ **do**
        $plainData \leftarrow RandomPermutation(table, plainData)$;
        $st_r \leftarrow (K \times (r + Nonce))\ mod\ B$
        $plainData \leftarrow reverse(plainData,\ st_r)$;
        $parityBit(plainData,\ st_r)$;
        $plainData \leftarrow leftRotation(plainData,\ st_r)$;
    end for
    return $plainData$;
end if
return -1

Algorithm 2 Sorting table function.

Require: (**Arg** *table*);

Ensure: *table*;

Temporal: *temp*;
for ($i \in [0, B-1]$) **do** ▷ `Sorting` *table* `depending on` V_i `values and moving their bit positions`
  for $j \in [i+1, B-1]$ **do**
    if (table[V][i] > table[V][j]) **then**
      $temp \leftarrow table[V][i]$;
      $table[V][i] \leftarrow table[V][j]$;
      $table[V][j] \leftarrow temp$;
      $temp \leftarrow table[orgPos][i]$;
      $table[orgPos][i] \leftarrow table[orgPos][j]$;
      $table[orgPos][j] \leftarrow temp$;
    end if
  end for
end for
for ($i \in [0, B-1]$) **do** ▷ `inserting the new bit positions`
  for $j \in [0, B-1]$ **do**
    if table[newPos][j] = i **then**
      $table[newPos][i] \leftarrow j$;
    end if
  end for
end for
return *table*

Proof. The integer $i = V_i \times B'$, where B' is a multiplicative inverse of a modulo K, is clearly a solution. For any integer i', we have $B \times i' \equiv V_i \pmod{K}$ if and only if $B \times i' \equiv B \times i \pmod{K}$ which holds if and only if $i \equiv i' \pmod{K}$.

Suppose that $B, V_i, K \in Z$ with $K > 0, B \neq 0$, and $gcd(B, K) = 1$. This theorem says that there exists unique integers i and j satisfying: $B \times i \equiv V_i \pmod{K}$, $B \times j \equiv V_j \pmod{K}$, and $V_i \neq V_j$ if $i \neq j$ with $0 \leq i, j < K$. In the proposed technique we avoid the value $i = 0$ to ensure a full bit permutation. □

We observe that any value of i that verifies $(B \times i) < K$, the congruence modulo K equals to $B \times i$ (i.e., $B \times i \equiv B \times i \pmod{K}$). This situation leads to generate some ordered V_i values. Following to the principle of the proposed technique, ordered values of V_i means encrypting data with weak permutation. For example, by using $K = 101$ and $B = 16$ on Figure 5, V_i will be $[16, 32, 48, 64, 80, 96, 11, 27, 43, 59, 75, 91, 6, 22, 38, 54]$ when we use $i = \{1, 2, \ldots 16\}$ in Equation (3). The values that are in ascendant order (here there are three waves of ordered values: [16 – 96], [11 – 91] and [6 – 54]) will minimize the randomization of the permutation phase of the encryption process. In order to avoid these values, we let run i from $K - 1$ until $K - B$ as shown in Equation (1). V_i values will then determine the bit permutation of plain data as shown by the example presented in Figure 5

An analysis shows that the total number of ordered compositions of V_i that can be obtained by using different values of K equals $B!$. Then, in the aim to reduce brute force attacks, the size of the data to be encrypted should be configured in such a way that $B! \approx +\infty$. As a result, the strength of the proposed technique is relatively proportional to the size of B in terms of bits, where there is no restriction concerning the preferred size of K. The computation task of checking $\gcd(K, B) = 1$ can be avoided if the used size of B is a prime number, case in which any value of K will verify $\gcd(K, B) = 1$. Choosing a prime number B will help us to minimize the computation and to avoid repute computation in case that $\gcd(K, B) \neq 1$. The process of sharing securely the same value of K between two or more devices can be reached by using one of the known algorithms, for instance the asymmetric key sharing algorithm or a pre-configured symmetric key sharing scheme. On the other hand, the number rnd of rounds is the second parameter. Choosing a large number of rounds presumably provides an increased level of security. We note here that the number of rounds depends on the desired security level and the supplied size of B. However, choosing a large number of rounds also implies a need for more memory and computation time. The analysis of the example shown in Figure 7 says that a brute force attack may find the original data from the used permutation at a maximum of 16! tests. The speed of growth of the factorial function offers a great advantage to reduce brute force attacks.

In the proposed technique we use two kinds of permutation: the first is by ordering the values of V_i and the second is achieved from the rotation. However, the technique still needs a substitution function to avoid frequency analysis attacks. Therefore, we have reversed all bits starting from st_r until the B-bit

position. Hence, the value of st_r can be exploited for multiple uses: it offers a random permutation in each round, ensures the bit substitutions and extracts the integrity bits. The integrity is ensured by computing the parity of bits before starting the permutation. In FlexenTech, the size of parity bits equals rnd and the receiver detects any change on one bit by comparing the parity bits of the obtained plain data with the right bit of the integrity bits. Otherwise, changing even bits will not be detected in the obtained plain data. This change can be detected in the next rounds. The integrity is ensured by satisfying the condition of preventing the bits change in case that an even number of changed bits are still in the same part, i.e., in the reversed bits part or in the other part. Hence, the number of rounds has a straight impact on data integrity strength.

Let P be the probability to make a change on some bits over the ciphered data that generate the same parity bits. The value of P is related to the size of parity bits, i.e., the number of rounds, and the value of $Nonce$.

In this technique, an attacker can falsify the data without any change at the parity bits level only when the falsified bits meet the following two conditions:

1. Falsifying an even number of bits situated in the reversed part, i.e., between $[st_r, B]$;
2. Each round contains an even number of falsified bits in the reversed part.

The first condition seems easy to be met, but each pair of falsified bits should be situated on the same side (either on the reversed or on the unreversed bits side) for every round. Hence, the second condition says that the data integrity of FlexenTech depends on the size of the parity bits, i.e., the number of rounds. Then, the probability P of an attacker to falsify e bits (e being an even number) of the ciphered data without a change at the parity bits is calculated as:

$$P^e_{st_r} = \frac{st_r}{B} \times \frac{st_r - 1}{B - 1} \times \ldots \frac{st_r - e - 1}{B - e - 1} + \frac{B - st_r}{B} \times \frac{B - st_r - 1}{B - 1} \cdots \times \frac{B - st_r - e - 1}{B - e - 1} \tag{4}$$

with total probability $P = P^e_1 \times P^e_2 \ldots P^e_{rnd}$.

Equation (4) says that there are two probabilities of successful falsifications. The first probability could be obtained when the falsification has been made over the unreversed side and these falsified bits are still on the same side for each round. The same holds for the second probability, but the falsification has been made over the reversed bits side. Hence, we observe that the number of rounds has an important impact on the probability to succeed in falsifying the ciphered data.

3.4. Discussion

We divide the discussion into two parts: 1) Choosing the adequate key establishment algorithm to securely compute the shared key (i.e., K). 2) The robustness and efficiency of this technique to cope with the most known attacks.

3.4.1. Key establishment algorithm

Key establishment is the most fundamental cryptographic primitive in all types of applications. However, the nature of such connected devices limits the use of conventional key establishment techniques. Many researchers have proposed to use symmetric key pre-distribution schemes for the key establishment objective [40, 42]. These schemes consider the resource limitation, the communication capability and the computation speed [43]. However, symmetric schemes often have a centralized aspect, focus on pre-configuration and define a pool key principle which leads us to exploit asymmetric key establishment schemes [44]. The authors of [45] benchmark some public key protocols that are usually used for a symmetric key agreement to determine the most appropriate for the requirements of critical infrastructure and emergency applications. Based on these algorithms, the analysis shows that there are remarkable performance benefits with the $Curve25519$, especially $Four\mathbb{Q}$ [46] which uses the *Elliptic Curve Diffie-Hellman* scheme.

In the proposed protocol, we use $Four\mathbb{Q}$ which is a new elliptic curve algorithm released by Microsoft Research in 2015. $Four\mathbb{Q}$ is not used yet in standard or known protocols. Technically, $Four\mathbb{Q}$ targets the 128 bit security level and its high performance is mainly obtained from the decomposition of the total number of the elliptic curve group operations and fast arithmetic modulo computation using the Mersenne prime $p = 2^{127} - 1$. In term of speed, $Four\mathbb{Q}$ uses the endomorphisms to accelerate scalar multiplications via four-dimensional decompositions, where the computation of scalar multiplications is significantly faster than all known curve-based cryptographic primitives.

3.4.2. FlexenTech's robustness and efficiency

During the development of the FlexenTech encryption technique, we began to focus our attention on how to find adequate solutions to the encountered IoT issues by taking into account the designed technique that should additionally provide a high-security level when suitable parameters are assigned. Our technique demonstrates its robustness and efficiency by providing data encryption, data integrity and data freshness, as will be explained below.

Encryption. The proposed technique converts a plain text into a ciphered text with the same text size B. Furthermore, the proposed technique considers permutation and substitution to avoid frequency cryptanalysis attacks. That means that the obtained ciphered text has no trace or information related to the original text. In computation and efficiency terms, the proposed technique needs only simple operations in order to encrypt a text with a given key K by computing B values of V_i and ordering them. In each round, the bits of the input text should be rotated starting from st_r, ordered with respect to V_i values and reversing the bits that are on the right side of st_r, see Figure 7(a). The key restriction here is that the value of K should be greater than B which offers a large pool of keys and ensures flexible key configurations. This encryption mechanism offers a pool of size $B! - B$.

Data integrity. The integrity approaches aim to detect the alteration in the data that will be sent. Any even slight modification should be of effect on the integrity bits. In the decryption phase, the authorized entity has to check a possible alternation by computing the parity bit of the bits that are listed on the right side of st_r, then compare it with the originally received bit in each round. This technique is powerful for detecting the integrity of data during the data decryption, i.e., we do not need to execute an additional separated task. Furthermore, it is possible to detect a data alteration before obtaining the plain text. In fact, this principle will offer a significant reduction in term of computation. On the other hand, such applications request to hide the integrity bits in the ciphered data which is easily reached in our proposed technique by computing the r next V_i values ($V_{r+1},\ V_{r+2} \dots V_{r+B}$) under the condition that $K > (r + B)$. The parity bits and the ciphered data can then be encrypted in an additional round.

3.5. Data freshness

The importance of data freshness is increasing, especially in the domain of distributed smart networks which are composed of a large-scale set of autonomous data sources. The use of the encrypted data with same freshness may lead to unauthorized access problems. In fact, data freshness is indispensable in Wireless Sensor Networks, IoT and other domains, to reduce various types of attacks such as replay attacks and extraction of some confidential information from the encrypted data. At the same time, we have to be interested in the overall generated overhead in order to ensure data freshness. In this paper, we propose an efficient technique to ensure the data freshness by using a $Nonce$ during the data encryption/decryption phases, where the used $Nonce$ has no restriction in term of size. The way to avoid the augmentation of the overhead is to introduce the $Nonce$ to compute the rotation in each round. In fact, by using this principle the following services can be ensured:

- Data freshness is ensured without any additional overhead. However, the entities publicly share the used $Nonce$.
- Encapsulating the whole encrypted data, that will be sent, allows to distinguish ciphered text from the same plain data by using a different $Nonce$ value in each case.
- A large counter size is useful to be used as a $Nonce$ in order to encrypt data with different counter values, i.e., for each secret K the counter can be used 2^x times, where x is the $Nonce$ size in terms of bits.

The data freshness offers many other services and advantages mainly in real-time applications including e-commerce, sensor data fusion, traffic control, and monitoring. To reach this aim, the data freshness process of this technique focuses on the communication speed, where no overhead is added. The proposed technique gives a configurable range of parameter values so that user devices may run the encryption algorithm

whose security and speed are accepted for their application. Unlike several encryption techniques, which have no parameterization and hence no flexibility, the proposed encryption algorithm permits upgrades as necessary. The choice of r affects both encryption speed and security. For some domains, real-time may be the most critical requirement. This kind of application looks for the best security obtainable within a given encryption time requirement. Choosing a small value of rnd (say $rnd = 4$) may provide some security, albeit modest, within the given speed constraint. The size overhead is critical, because it immediately affects the communication speed and the lifetime of connected devices, as well as the packet loss rate caused by packet collisions.

4. Implementation and results

This section gives the experimental setup details used in the evaluation of FlexenTechit's reconfiguration flexibility. FlexenTech uses a pseudo random number as a key for encryption and decryption. It completes the rnd rounds of encryption on each B bits block of data. In all rounds, encryption is done using a function that ensures an irregular rotation. Furthermore, each round increases the number of integrity bits. With any unfixed parameter, the encryption strength of FlexenTech is directly tied to the predefined parameters during its execution. FlexenTech gives its users the ability to balance the weighing between encryption computational complexity, security level, and data load; specifically, users can adjust encryption speed, key setup time, and code size to balance performance. This makes FlexenTech suitable for application on resource constrained embedded devices.

FlexenTech is easy to implement efficiently in most common programming languages. For evaluation purposes, FlexenTech was coded in C (source code available from[2]). This implementation follows accurately the specifications given in Section 3.2 to generate the encryption example shown in Figure 7. The performance analysis evaluates the effect of the rnd and B parameters as a function of the computation time. Since the study relies on the development of a lightweight encryption technique for IoT systems, the proposed technique will be implemented and verified in two environments with the following specifications:

Raspberry Pi 3 Model B :

- CPU: Quad-Core 1.2GHz Broadcom BCM2837 64bit, ARM Cortex-A53 (ARMv8)
- RAM 1GB
- Raspbian is the official operating system
- Geany GCC c/c++ compiler

Laptop Acer Aspire E1-571 :

[2]https://github.com/smedileh/FlexenTech/blob/master/FlexenTech.c

- CPU: Intel Core i5-3230M 3rd Gen, 2.60 GHz Dual-core, HM77 Express
- RAM 8GB
- Windows 10 operating system
- Geany GCC c/c++ compiler

In the following subsections, we describe each experiment and present and analyse its results.

4.1. Number of Round

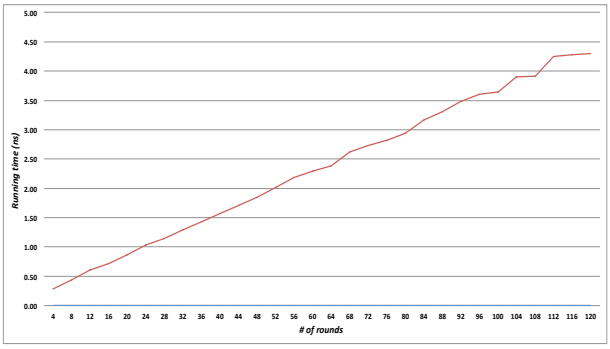
This experiment investigates the impact of changing rnd on FlexenTech's performance. Figures 8(a) and 8(c) plots rnd against the running time on both the Raspberry Pi device and the Acer laptop. The block size is fixed to 16 through all runs. As observed from Figure 8(a), when $rnd = 4$ the running time in the laptop does not exceed 0.25 ns. Hereafter, when rnd is increased to 8, the running time is propoertioally doubled to reach 0.5 ns. The same trend is observed consistently with other parameters, meaning that the time increases approximately by a factor of two. Similar performance is observed on the Raspberry Pi device. Figure 8(c) shows that the running time on the Raspberry Pi device continues to increase as rnd increases.

4.2. Block Size

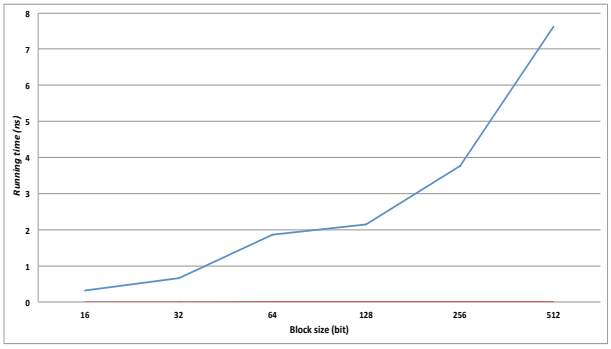
The experimental results illustrated in Figures 8(b) and 8(d) study the impact of the used block size on the encryption time and complexity. In these experiments, the rnd is fixed to 16 while varying the block size to 16, 32, 64, 128, 256 and 512. Figure 8(b) shows that, when the block size is as small as 16 bits, the running time does not exceed 0.35 ns. When the block size is fixed to 32 and 64 $bits$ the running time increases by a factor of 2 with a time of about 0.75 and 1.85 ns, respectively. When the block size is set to 128 $bits$ the running time does not increase as much and stands at 2.1 ns. However, when the block size increases to 256 and 512 $bits$, the running time increases significantly to about 180%. This increase in the running time is caused by the time consumed by the sorting function in Algorithm 1 (see line 9). Figure 8(d) shows similar performance is on the Raspberry Pi.

4.3. Comparative Performance Analysis

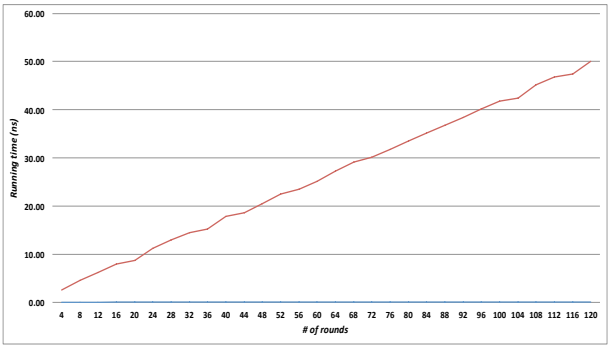
In this section we compare FlexenTech's encryption time with some common symmetric encryption techniques, namely FlexenTech, RC5, Blowfish and AES. Figure 9 shows the encryption speed of FlexenTech against these symmetric techniques running on the Acer laptop. We observe that FlexenTech outperforms both AES and Blowfish, and performs comparably to RC5 and FlexenTech. Similar to FlexenTech, RC5 and FlexenTech are simple algorithms with low memory requirements, which makes them suitable for resource constrained hardware.



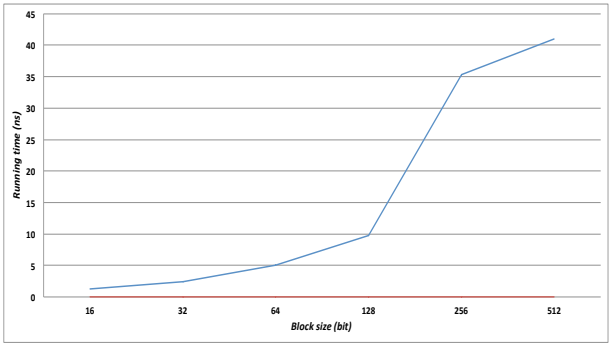
(a) Running time of the personal computer as a function of the number of rounds.



(b) Running time of the personal computer as a function of the block size.



(c) Running time of the Raspberry Pi device as a function of the number of rounds.



(d) Running time of the Raspberry Pi device as a function of the block size.

Figure 8: Running time as a function of the number of rounds and the block size.

Figures 10 and 11 show the encryption processing time on a Raspberry Pi device. We observe in Figure 10 that FlexenTech is faster than RC5 while providing good security level when using block size smaller than 64 bits, whereas RC5 provides security only if predefined suitable parameters are chosen. The security of RC5 depends on the size of the used key, which is stronger for a larger key size. Hence, FlexenTech is more efficient and flexible in term of the used block size.

We also studied the encryption processing time as a function of the number of rounds. Figure 11 shows that RC5 performs better if $rnd > 8$. However, FlexenTech still gives a reasonable encryption processing time even if $rnd < 32$. Due to the random permutations at the bit level of the plaintext, it is possible to use FlexenTech with one round to encrypt securely. Therefore, 4 rounds are largely enough in FlexenTech, whereas 12 rounds are required in RC5.

In terms of reconfiguration flexibility, Simon and Speck cryptographic schemes are designed with leveled flexibility allowing some different key and block sizes. Speck contains 10 variants where the block size is 32, 48, 64, 96 or 128 bits, and the key size is 64, 72, 96, 128, 144, 192 or 256 bits and the number of rounds depends on the parameters selected. Speck gets its nonlinearity from the modular addition operation; it has been demonstrated that key lengths below 80 bits do not provide a high level of security. When SIMON is executed on an 8-bit AVR microcontroller, Speck encryption with 64-bit blocks and a 128-bit key consumes 192 bytes of the Flash memory, temporary variables consume 112 bytes of RAM, takes 164 cycles to encrypt each byte in the block, and the *Attacked Rounds/Total Rounds* percentage ranges between 53% and 74%; however, reduced-round variants have been successfully compromised.

The Tiny Encryption Algorithm (TEA) cipher focuses on Feistel iterations by using 64-bit size blocks (32 bit words) and a 128-bit key (divided into 4 parts) where odd rounds use $K[0;1]$ and even rounds use $K[2;3]$. With the use of 32 cycles (64 rounds), the designers of TEA argue that 16 cycles may suffice but they suggest 32 cycles with the use of the constant "C" (used to prevent simple attacks based on the symmetry of the rounds). There are successful attacks on TEA with 17 rounds because TEA is susceptible to related-key attack which arise from the simplicity of its key schedule [47].

We summarise the performance of FlexenTech against of theses ciphers in Table 3. Overall, the security performance of block ciphers depends on the used block size, key size and number of rounds. Other encryption techniques prefer a large block size for faster execution due to the time spend in the cipher initialisation. Moreover, large keys result in lower computation time, because all bits of the key are involved in an execution cycle of the encryption. FlexenTech offers a major advantage by allowing efficient encryption using only one single round. Another advantage of FlexenTech is its reduced complexity, i.e., the key has two parts where the first is static and used once for the random permutation for all rounds, and the second is dynamic requiring its specific computation in each round to compute the start bit rotation. The static part of the key reduces computation cost, memory utilisation and energy consumption, which is of particular importance for the low-end IoT devices.

Technique	**Configuration**			**Performance**		
	Block size	**Key size**	**# of rounds**	**Computation time**	**Energy consumption**	**Against attacks**
FlexenTech	Variable	Variable	Open	Speed with such configuration	Low energy consumption	Brute force; Replay attack
RC5	32,64,128 bits	0–2040 bits	12 rounds	Reasonable time	Low with small # of rounds [48]	Differential attack
TEA	64 bits	128 bits	6–64 rounds	Speed with small # of rounds	Low energy consumption	Equivalent key attack
SIMON	64–256 bits	32–128 bits	32–72 rounds	Suitable time [49]	Moderate energy consumption	Weak keys
Blowfish	64 bits	32–448 bits	16 rounds	Moderate time [50]	High energy consumption [48]	Known plaintext
DES	64 bits	56 bits	16 rounds	Relatively slow [50]	Low with small block	Brute Force Attack
AES	128 bits	128, 192, 256 bits	Tied to the key size [49]	Relatively slow	Relatively low with small block [51]	Known plaintext

Security	**Standard encryption**		**Lightweight encryption**			
	RC5 [52, 28, 39, 53]	**Blowfish** [52, 54, 55]	**FlexenTech**	**TEA** [56, 52, 39, 53]	**SIT** [27]	**SIMON** [25, 53]
Confidentiality	★★★★	★★★★	★★★★	★★★	★★★	★★★
Authentication	★	★	★★	★★★	★★★	★★
Integrity	★★★	★★	★★★	★★	★★	★★
Authorization	★	★	★	★★	★	★★
Freshness	★★	★★	★★★	★	★★	★★

Table 3: Summary table of encryption techniques with various configurations

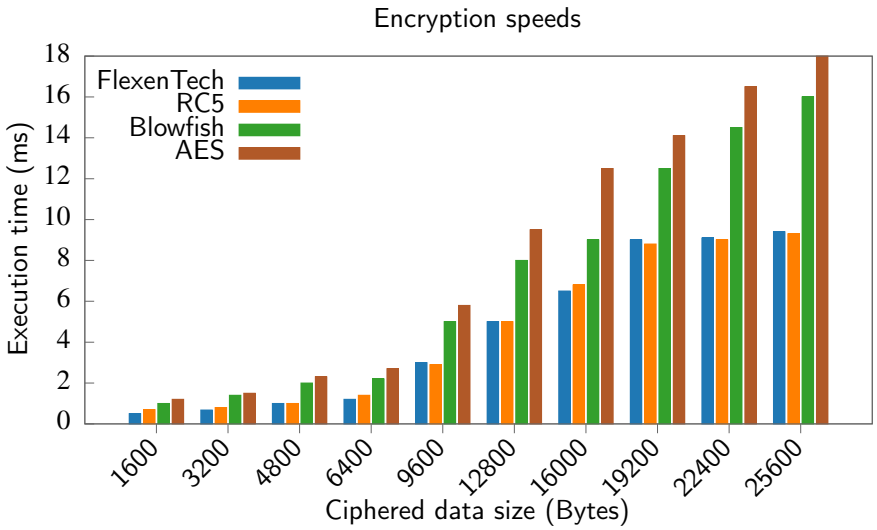


Figure 9: The execution speed of FlexenTech compared with other encryption techniques.

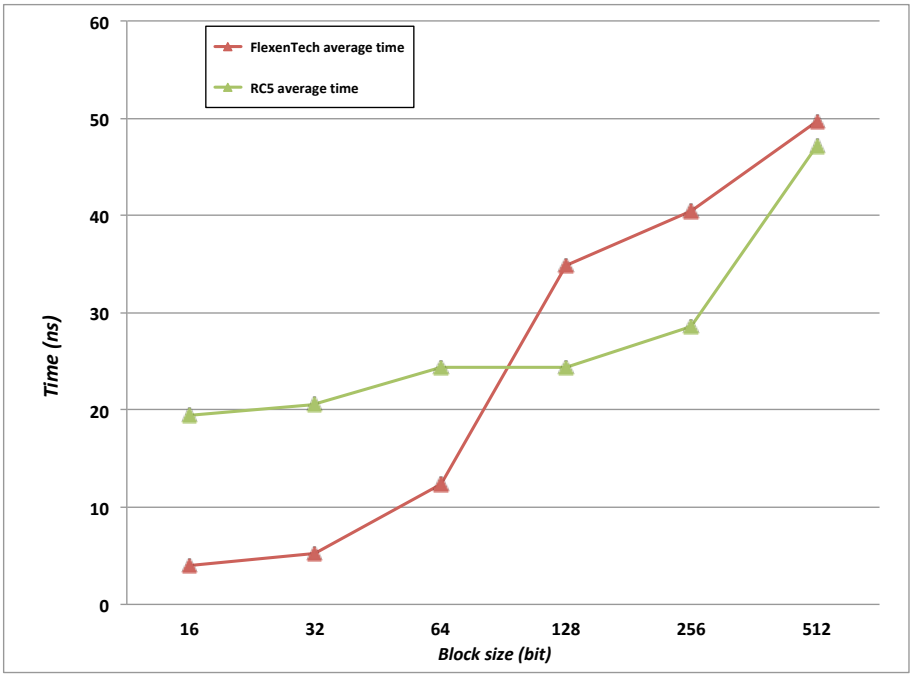


Figure 10: The execution time as a function of the block size.

5. Conclusion and future work

Nowadays, the main security challenge in the IoT environment is the design of efficient and lightweight encryption techniques in regard to resource limitations of connected devices. In this paper, we have proposed the FlexenTech encryption technique to cope with the device limitation challenge in order to render

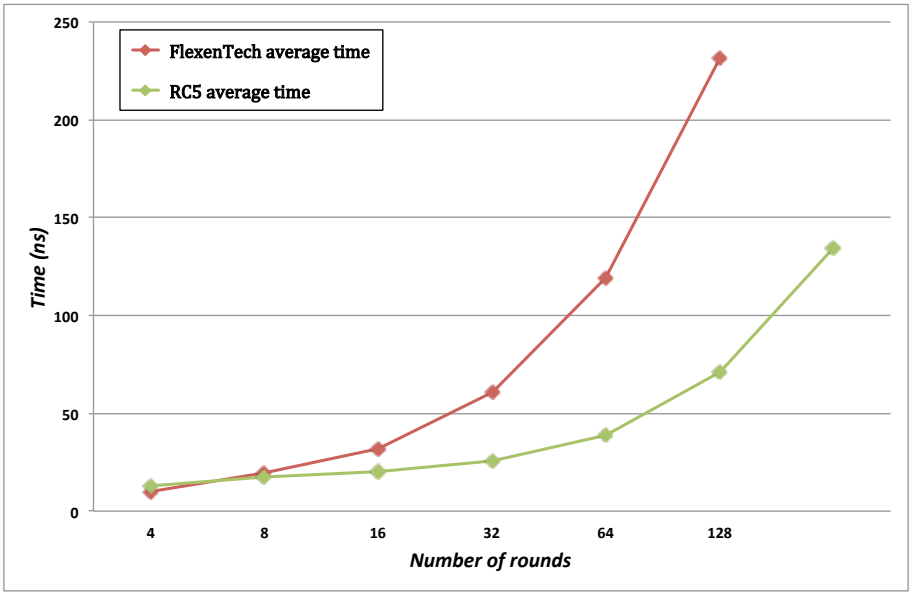


Figure 11: The execution time as a function of the number of rounds.

these devices more applicable in terms of security requirements. The mechanism of the proposed technique reduces the computation time incurred by data transmission, because we have minimized the number of rounds used to cipher the information and introduced many improvements to minimize computation. Reducing computation time offers a remarkable benefit in term of energy consumption of the autonomous IoT devices [57]. The implementation of FlexenTech upon low-end IoT devices shows its practical flexibility. This flexibility enables the choice of a minimal number of rounds and a smaller overhead size which leads to an improvement in encryption time without reducing security levels. The obtained enhancement on the encryption time decreases many risks of such attacks. A security analysis of the proposed technique also shows its configuration flexibility and feasibility in terms of limited resource consumption and low execution time.

In future research, we intend to evaluate FlexenTech on various IoT hardware platforms in different application scenarios that require various levels of security. This allows studying the configuration flexibility and security of FlexenTech under real-life conditions exposing potential practical technical issues that may hinder its application. In parallel to practical evaluation, we plan to formally model and analyse FlexenTech to prove its properties using SPIN model checker or similar tool. Another interesting future work avenue is to investigate the possibility to adapt this principle of FlexenTech to the use for cloud storage.

References

[1] M. Hammoudeh and R. Newman, "Information extraction from sensor networks using the watershed transform algorithm," *Information Fusion*, vol. 22, pp. 39 – 49, 2015. [Online]. Available: http://www.sciencedirect.com/science/

article/pii/S1566253513000808

[2] I. Kertiou, S. Benharzallah, L. Kahloul, M. Beggas, R. Euler, A. Laouid, and A. Bounceur, "A dynamic skyline technique for a context-aware selection of the best sensors in an iot architecture," *Ad Hoc Networks*, vol. 81, pp. 183–196, 2018.

[3] V. Hassija, V. Chamola, V. Saxena, D. Jain, P. Goyal, and B. Sikdar, "A survey on iot security: application areas, security threats, and solution architectures," *IEEE Access*, vol. 7, pp. 82 721–82 743, 2019.

[4] A. Laouid, A. Dahmani, A. Bounceur, R. Euler, F. Lalem, and A. Tari, "A distributed multi-path routing algorithm to balance energy consumption in wireless sensor networks," *Ad Hoc Networks*, vol. 64, pp. 53–64, 2017.

[5] M. Gulzar and G. Abbas, "Internet of things security: A survey and taxonomy," in *2019 International Conference on Engineering and Emerging Technologies (ICEET)*. IEEE, 2019, pp. 1–6.

[6] M. Patel and J. Wang, "Applications, challenges, and prospective in emerging body area networking technologies," *IEEE Wireless communications*, vol. 17, no. 1, 2010.

[7] B. L. P. Gassend, "Physical random functions," Ph.D. dissertation, Massachusetts Institute of Technology, 2003.

[8] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, and J. Schmidhuber, "Modeling attacks on physical unclonable functions," in *Proceedings of the 17th ACM Conference on Computer and Communication Security*. ACM, 2010, pp. 237–249.

[9] L. Ducas, V. Lyubashevsky, and T. Prest, "Efficient identity-based encryption over ntru lattices," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2014, pp. 22–41.

[10] C. E. Shannon, "Communication theory of secrecy systems," *Bell system technical journal*, vol. 28, no. 4, pp. 656–715, 1949.

[11] B. Preneel and T. Takagi, *Cryptographic Hardware and Embedded Systems–CHES 2011: 13th International Workshop, Nara, Japan, September 28–October 1, 2011, Proceedings*. Springer, 2011, vol. 6917.

[12] P. Gutmann, D. Naccache, and C. C. Palmer, "When hashes collide [applied cryptography]," *IEEE security & privacy*, vol. 3, no. 3, pp. 68–71, 2005.

[13] S. Belguith, N. Kaaniche, M. Hammoudeh, and T. Dargahi, "Proud: verifiable privacy-preserving outsourced attribute based signcryption supporting access policy update for cloud assisted iot applications," *Future Generation Computer Systems*, 2019.

[14] J. Carey, "Protection of personal data contained on an rfid-enabled device," Sep. 11 2018, uS Patent App. 15/445,328.

[15] N. Oualha and K. T. Nguyen, "Lightweight attribute-based encryption for the internet of things," in *Computer Communication and Networks (ICCCN), 2016 25th International Conference on*. IEEE, 2016, pp. 1–6.

[16] X. Yao, Z. Chen, and Y. Tian, "A lightweight attribute-based encryption scheme for the internet of things," *Future Generation Computer Systems*, vol. 49, pp. 104–112, 2015.

[17] M. AlShaikh, L. Laouamer, L. Nana, and A. C. Pascu, "Efficient and robust encryption and watermarking technique based on a new chaotic map approach," *Multimedia Tools and Applications*, vol. 76, no. 6, pp. 8937–8950, 2017.

[18] H. Luo, G. Wen, J. Su, and Z. Huang, "Slap: Succinct and lightweight authentication protocol for low-cost rfid system," *Wireless Networks*, vol. 24, no. 1, pp. 69–78, 2018.

[19] Z. Liu, D. Liu, L. Li, H. Lin, and Z. Yong, "Implementation of a new rfid authentication protocol for epc gen2 standard," *IEEE Sensors Journal*, vol. 15, no. 2, pp. 1003–1011, 2015.

[20] M. Hammoudeh, G. Epiphaniou, S. Belguith, D. Unal, B. Adebisi, T. Baker, A. S. M. Kayes, and P. Watters, "A service-oriented approach for sensing in the internet of things: Intelligent transportation systems and privacy use cases," *IEEE Sensors Journal*, pp. 1–1, 2020.

[21] J. Hosseinzadeh and M. Hosseinzadeh, "A comprehensive survey on evaluation of lightweight symmetric ciphers: hardware and software implementation," *Advances in Computer Science: an International Journal*, vol. 5, no. 4, pp. 31–41, 2016.

[22] D. Engels, M.-J. O. Saarinen, P. Schweitzer, and E. M. Smith, "The hummingbird-2 lightweight authenticated encryption

algorithm," in *International Workshop on Radio Frequency Identification: Security and Privacy Issues*. Springer, 2011, pp. 19–31.

[23] W. Wu and L. Zhang, "Lblock: a lightweight block cipher," in *International Conference on Applied Cryptography and Network Security*. Springer, 2011, pp. 327–344.

[24] D. E. Kouicem, A. Bouabdallah, and H. Lakhlef, "Internet of things security: A top-down survey," *Computer Networks*, 2018.

[25] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers, "Simon and speck: Block ciphers for the internet of things." *IACR Cryptology ePrint Archive*, vol. 2015, p. 585, 2015.

[26] M. Kumar, S. Kumar, R. Budhiraja, M. Das, and S. Singh, "Lightweight data security model for iot applications: A dynamic key approach," in *Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), 2016 IEEE International Conference on*. IEEE, 2016, pp. 424–428.

[27] M. Usman, I. Ahmed, M. I. Aslam, S. Khan, and U. A. Shah, "Sit: a lightweight encryption algorithm for secure internet of things," *arXiv preprint arXiv:1704.08688*, 2017.

[28] S. Singh, P. K. Sharma, S. Y. Moon, and J. H. Park, "Advanced lightweight encryption algorithms for iot devices: survey, challenges and solutions," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–18, 2017.

[29] R. Montero-Canela, E. Zambrano-Serrano, E. I. Tamariz-Flores, J. M. Muñoz-Pacheco, and R. Torrealba-Meléndez, "Fractional chaos based-cryptosystem for generating encryption keys in ad hoc networks," *Ad Hoc Networks*, vol. 97, p. 102005, 2020.

[30] S. Belguith, N. Kaaniche, and G. Russello, "Pu-abe: Lightweight attribute-based encryption supporting access policy update for cloud assisted iot," in *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*. IEEE, 2018, pp. 924–927.

[31] S. Moffat, M. Hammoudeh, and R. Hegarty, "A survey on ciphertext-policy attribute-based encryption (cp-abe) approaches to data security on mobile devices and its application to iot," in *Proceedings of the International Conference on Future Networks and Distributed Systems*. ACM, 2017, p. 34.

[32] S. Al Salami, J. Baek, K. Salah, and E. Damiani, "Lightweight encryption for smart home," in *2016 11th International Conference on Availability, Reliability and Security (ARES)*. IEEE, 2016, pp. 382–388.

[33] J. Li, Y. Zhang, J. Ning, X. Huang, G. S. Poh, and D. Wang, "Attribute based encryption with privacy protection and accountability for cloudiot," *IEEE Transactions on Cloud Computing*, 2020.

[34] S. Belguith, N. Kaaniche, G. Russello *et al.*, "Lightweight attribute-based encryption supporting access policy update for cloud assisted iot," in *Proceedings of the 15th International Joint Conference on e-Business and Telecommunications-Volume 1: SECRYPT*. SciTePress, 2018, pp. 135–146.

[35] O. Alphand, M. Amoretti, T. Claeys, S. Dall'Asta, A. Duda, G. Ferrari, F. Rousseau, B. Tourancheau, L. Veltri, and F. Zanichelli, "Iotchain: A blockchain security architecture for the internet of things," in *2018 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2018, pp. 1–6.

[36] L. Seitz, G. Selander, E. Wahlstroem, S. Erdtman, and H. Tschofenig, "Authentication and authorization for constrained environments (ace)," *Internet Engineering Task Force*, 2017.

[37] M. Vučinić, B. Tourancheau, F. Rousseau, A. Duda, L. Damon, and R. Guizzetti, "Oscar: Object security architecture for the internet of things," *Ad Hoc Networks*, vol. 32, pp. 3–16, 2015.

[38] D. Dinu, Y. Le Corre, D. Khovratovich, L. Perrin, J. Großschädl, and A. Biryukov, "Triathlon of lightweight block ciphers for the internet of things," *Journal of Cryptographic Engineering*, pp. 1–20, 2015.

[39] S. Rajesh, V. Paul, V. G. Menon, and M. R. Khosravi, "A secure and efficient lightweight symmetric encryption scheme for transfer of text files between embedded iot devices," *Symmetry*, vol. 11, no. 2, p. 293, 2019.

[40] A. Laouid, A. Dahmani, H. R. Hassen, A. Bounceur, R. Euler, F. Lalem, and A. Tari, "A self-managing volatile key scheme for wireless sensor networks," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–16, 2018.

[41] V. Shoup, *A computational introduction to number theory and algebra*. Cambridge university press, 2009.

[42] A. Laouid, M.-L. Messai, A. Bounceur, R. Euler, A. Dahmani, and A. Tari, "A dynamic and distributed key management scheme for wireless sensor networks," in *Proceedings of the International Conference on Internet of things and Cloud Computing*. ACM, 2016, p. 70.

[43] C.-Y. Chen and H.-C. Chao, "A survey of key distribution in wireless sensor networks," *Security and Communication Networks*, vol. 7, no. 12, pp. 2495–2508, 2014.

[44] S.-H. Seo, J. Won, S. Sultana, and E. Bertino, "Effective key management in dynamic wireless sensor networks," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 2, pp. 371–383, 2015.

[45] R. Alvarez, C. Caballero-Gil, J. Santonja, and A. Zamora, "Algorithms for lightweight key exchange," *Sensors*, vol. 17, no. 7, p. 1517, 2017.

[46] C. Costello and P. Longa, "Four$\mathbb{Q}$: Four-dimensional decompositions on a $\mathbb{Q}$-curve over the mersenne prime," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2015, pp. 214–235.

[47] S. Hong, D. Hong, Y. Ko, D. Chang, W. Lee, and S. Lee, "Differential cryptanalysis of tea and xtea," in *International Conference on Information Security and Cryptology*. Springer, 2003, pp. 402–417.

[48] N. R. Potlapally, S. Ravi, A. Raghunathan, and N. K. Jha, "Analyzing the energy consumption of security protocols," in *Proceedings of the 2003 international symposium on Low power electronics and design*, 2003, pp. 30–35.

[49] N. Alassaf, A. Gutub, S. A. Parah, and M. Al Ghamdi, "Enhancing speed of simon: a light-weight-cryptographic algorithm for iot applications," *Multimedia Tools and Applications*, vol. 78, no. 23, pp. 32 633–32 657, 2019.

[50] T. Nie, C. Song, and X. Zhi, "Performance evaluation of des and blowfish algorithms," in *2010 International Conference on Biomedical Engineering and Computer Science*. IEEE, 2010, pp. 1–4.

[51] P. Prasithsangaree and P. Krishnamurthy, "Analysis of energy consumption of rc4 and aes algorithms in wireless lans," in *GLOBECOM'03. IEEE Global Telecommunications Conference (IEEE Cat. No. 03CH37489)*, vol. 3. IEEE, 2003, pp. 1445–1449.

[52] M. Dener, "Security analysis in wireless sensor networks," *International Journal of Distributed Sensor Networks*, vol. 10, no. 10, p. 303501, 2014.

[53] S. S. Dhanda, B. Singh, and P. Jindal, "Lightweight cryptography: A solution to secure iot," *Wireless Personal Communications*, pp. 1–34, 2020.

[54] M. Suresh and M. Neema, "Hardware implementation of blowfish algorithm for the secure data transmission in internet of things," *Procedia technology*, vol. 25, pp. 248–255, 2016.

[55] M. N. A. Wahid, A. Ali, B. Esparham, and M. Marwan, "A comparison of cryptographic algorithms: Des, 3des, aes, rsa and blowfish for guessing attacks prevention," *Journal Computer Science Applications and Information Technology*, vol. 3, pp. 1–7, 2018.

[56] D. J. Wheeler and R. M. Needham, "Tea, a tiny encryption algorithm," in *International Workshop on Fast Software Encryption*. Springer, 1994, pp. 363–366.

[57] Z. Sayah, O. Kazar, B. Lejdel, A. Laouid, and A. Ghenabzia, "An intelligent system for energy management in smart cities based on big data and ontology," *Smart and Sustainable Built Environment*, 2020.